CLAIMS

1. In a Java computing environment, a Java macro instruction representing:

a sequence of Java Bytecode instructions consisting of a Java instantiation Bytecode instruction immediately followed by a Java Duplicate the stack Bytecode instruction,

wherein said Java macro instruction can be executed by a Java virtual machine operating in said Java computing environment, and

wherein, when said Java macro instruction is executed, the operations that are performed by said conventional sequence of Java Bytecode instructions are performed.

2. A Java macro instruction as recited in claim 1, wherein said Java macro instruction consists of a conventional Java instantiation Bytecode instruction immediately followed by a conventional Java Duplicate the stack Bytecode instruction.

3. A Java macro instruction as recited in claim 1, wherein said Java macro instruction is generated during the Java Bytecode verification phase.

4. A Java macro instruction as recited in claim 1, wherein said Java virtual machine internally represents Java instructions as a pair of streams.

5. A Java macro instruction as recited in claim 4,

wherein said pair of streams includes a code stream and a data stream,

wherein said code stream is suitable for containing a code portion of said Java macro instruction, and

wherein said data stream is suitable for containing a data portion of said Java macro instruction.

6. A Java macro instruction as recited in claim 5,

wherein said Java macro instruction is generated only when said virtual machine determines that said Java macro instruction should replace said conventional sequence.

7. A Java macro instruction as recited in claim 6, wherein said determination is made based on a predetermined criteria.

8. A Java macro instruction as recited in claim 7, wherein said predetermined criteria is whether said conventional sequence has been repeated more than a predetermined number of times.

9. In a Java computing environment, a Java macro instruction representing:

a sequence of Java Bytecode instructions consisting of an inventive Java New Bytecode instruction immediately followed by an inventive Java Dup Bytecode instruction,

wherein said Java macro instruction can be executed by a Java virtual machine operating in said Java computing environment, and

wherein, when said Java macro instruction is executed, the operations that are performed by said sequence of Java Bytecode instructions are performed.

10. A Java macro instruction as recited in claim 9,

wherein said Java Dup Bytecode instruction represents a virtual machine instruction suitable for execution in a virtual machine to duplicate values stored in an execution stack on top of said execution stack,

wherein said Dup virtual machine instruction represents two or more conventional Java Bytecode executable instructions that are also suitable for duplicating values stored in the execution stack on top of the execution stack.

11. A Java macro instruction as recited in claim 10, wherein values that can be duplicated on the execution stack are not limited to values that are within the first, second, and third positions from the top of the stack.

12. A Java macro instruction as recited in claim 11, wherein the values duplicated on top of the stack can be 4 byte values or 8 byte values.

13. A Java macro instruction as recited in claim 12, wherein said virtual machine instruction has a parameter associated with it to indicate which value stored in the execution stack should be duplicated on the top of the stack.

14. A Java macro instruction as recited in claim 9, wherein said Java New Bytecode instruction represents a virtual machine instruction suitable for execution in a virtual machine to instantiate Java objects and arrays, the virtual machine instruction representing two or more Java Bytecode executable instructions that are also suitable for instantiation of Java objects or arrays.

15. A Java macro instruction as recited in claim 14, wherein the instantiation of Java objects and arrays are performed by determining the type of the object or array based on a parameter that is associated with the object or array.

16. A computer readable media including computer program code for a Java macro instruction, said Java macro instruction representing:

a sequence of Java Bytecode instructions consisting of an inventive Java New Bytecode instruction immediately followed by an inventive Java Dup Bytecode instruction,

wherein said Java macro instruction can be executed by a Java virtual machine operating in said Java computing environment, and

wherein, when said Java macro instruction is executed, the operations that are performed by said sequence of Java Bytecode instructions are performed.

17. A computer readable media as recited in claim 16,

wherein said Java Dup Bytecode instruction represents a virtual machine instruction suitable for execution in a virtual machine to duplicate values stored in an execution stack on top of said execution stack, and

wherein said Dup virtual machine instruction represents two or more conventional Java Bytecode executable instructions that are also suitable for duplicating values stored in the execution stack on top of the execution stack.

18. A computer readable media as recited in claim 17, wherein values that can be duplicated on the execution stack are not limited to values that are within the first, second, and third positions from the top of the stack.

19. A computer readable media as recited in claim 17, wherein said virtual machine instruction has a parameter associated with it to indicate which value stored in the execution stack should be duplicated on the top of the stack.

20. A computer readable media as recited in claim 17, wherein said Java New Bytecode instruction represents a virtual machine instruction suitable for execution in a virtual machine to instantiate Java objects and arrays, the virtual machine instruction representing two or more Java Bytecode executable instructions that are also suitable for instantiation of Java objects or arrays.

21. A computer readable media as recited in claim 20, wherein the instantiation of Java objects and arrays are performed by determining the type of the object or array based on a parameter that is associated with the object or array.